Case Study: Test Results of a Tool and Method for In-Flight, Adaptive Control System Verification on a NASA F-15 Flight Research Aircraft

Stephen Jacklin and Johann Schumann
NASA Ames Research Center, Moffett Field, CA 94035

John Bosworth, Peggy Williams-Hayes, and Richard Larson
NASA Dryden Flight Research Center, Edwards, CA 93523

Abstract

This paper presents the Confidence Tool developed at NASA Ames Research Center as a step towards providing a method to provide a metric to monitor adaptive learning algorithm performance in operating conditions for which the correct learning solution is unknown. These control systems primarily comprise learning systems utilizing neural network models, that use large over-parameterized models where the true system model is unknown because of uncertainties and un-modeled parameters. The Confidence Tool is a dynamic monitor which checks all input and output values of the neural network and determines if the result of the neural network is reliable. During the operation of the system (i.e., during the flight), a metric called the confidence measure is calculated for the outputs of the neural network. This metric is based upon a statistical model of the system. This paper presents the manner in which the confidence tool can be integrated into an adaptive system, a description of the Confidence Tool, and finally a comparison of simulation and flight test data.

Introduction

Adaptive control technologies that incorporate learning algorithms have been proposed to enable autonomous flight control and to maintain vehicle performance in the face of unknown, changing, or poorly defined operating environments.[1-2] At the present time, however, it is unknown how adaptive algorithms can be routinely verified, validated, and certified for use in safety-critical applications. Rigorous methods for adaptive software verification and validation must be developed to ensure that the control software functions as required and is highly safe and reliable.

A large gap appears to exist between the point at which control system designers feel the verification process is complete, and when FAA certification officials agree it is complete. Certification of adaptive flight control software is complicated by the use of learning algorithms (e.g., neural networks) and varying degrees of system non-determinism. Analytical efforts must be made in the verification process to place guarantees on learning algorithm stability, rate of convergence, and convergence accuracy. To satisfy FAA certification requirements, it must also be demonstrated that the adaptive flight control system is also able to fail and still allow the aircraft to be flown safely or to land, while at the same time providing a means of crew notification of the (impending) failure. It was for this purpose that the NASA Ames Confidence Tool was developed.[3]

This paper presents the application of the Confidence Tool as a means of providing a metric to monitor adaptive learning algorithm performance in operating conditions for which the correct learning solution is unknown. As a preliminary step, the paper will first describe the type of learning system under discussion. Then the Confidence Tool will be presented, followed by a comparison of simulation and flight test data.

Adaptive Flight Control System Architectures

Control systems for large aircraft and some spacecraft are often comprised of hybrid systems involving both inner-loop and outer-loop control architectures (Fig. 1). The outer-loop or executive controller

Presented at the 7th World Congress on Computational Mechanics
July 16-21, 2006, Los Angeles, CA

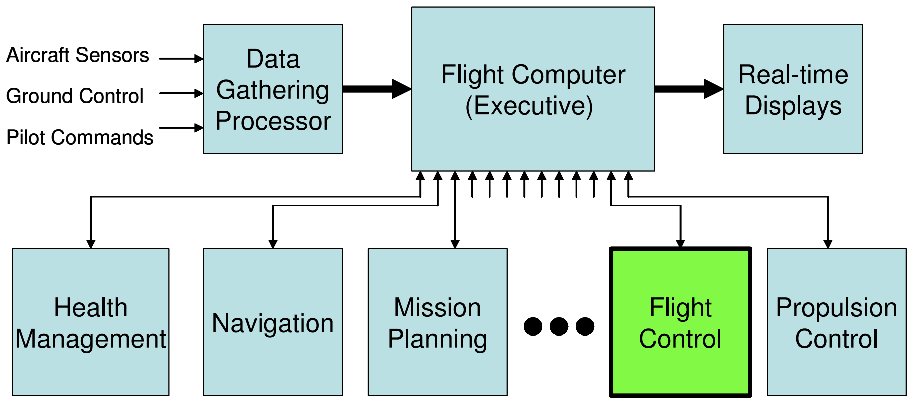

Figure 1. Flight controller architecture.

governs the conditional execution of several inner loop controllers. The outer-loop controller may coordinate a myriad of mission management actions including providing a human-machine interface, health monitoring, engine performance monitoring, navigation, guidance, and flight control. Outer-loop control software is generally comprised of finite state automata in which the sequencing of control tasks and multiple program threads is done conditionally based on finite state logic. The lower-level processes may be either finite state or continuous systems. The majority of inner-loop flight control systems require use of continuous variables and are most commonly represented as a coupled set of first order differential equations:

$$\begin{aligned} \frac{dx}{dt} &= Ax + Bu \\ y &= Cx + Du \end{aligned} \qquad (1)$$

where x is the dynamic state of the vehicle (rates, positions), u is a vector of control signals, and y is a vector of system measurements. A and B are generally matrices. Figure 2 illustrates a typical closed-loop flight control representation. In this figure, sensors are used to deduce the state of the aircraft for direct comparison to the desired state. The difference is referred to as the error signal. A PID (proportional, integral, derivative) controller may then be used to form control signals that are proportional to either the error itself, the integral of the error signal, and/or the derivative of the error signal:

$$Control = K_P(error) + K_I\left(\int(error)dt\right) + K_D\left(\frac{d}{dt}(error)\right) \qquad (2)$$

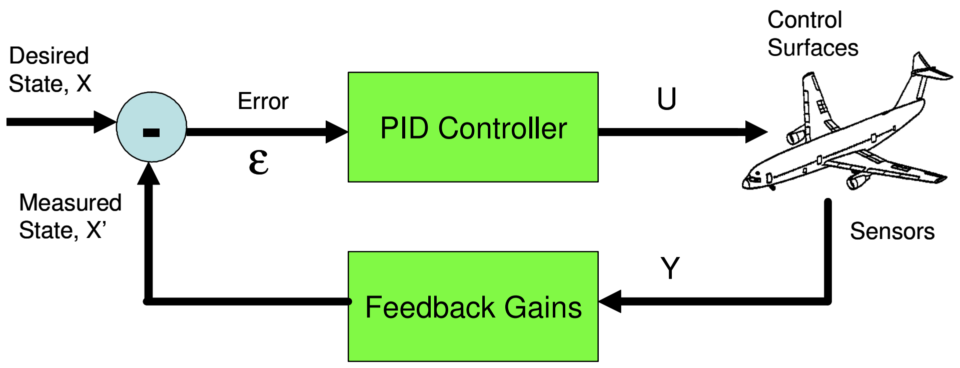

Figure 2. PID inner-loop controller.

The proportionality constants (K_P, K_I and K_D) are called the gains of the system and are vectors for multi-input, multi-output systems. Tuning the controller is a matter of adjusting the gain values to achieve optimal performance. Generally, there is a range of gain values that yield acceptable controller performance. However, when the gains are selected too large, the system may be over-controlled and exhibit instability. Similarly, if the gain values are selected too low, the system response is usually made to become too sluggish, sometimes even to the point of instability. Once the optimal values for the gains have been found, they are used as fixed constants in the control law. For this reason, this type of controller is often referred to as a “fixed-gain” controller.

Fixed-gain controllers have the limitation that the controller gains cannot be changed once the controller is placed into operation. Unforeseen events, such as sudden loss of a control surface or even the gradual deterioration of control system components, may render the fixed-gain control system unusable. This is because the control system was designed to operate under the assumption of all control surfaces being fully functional

Adaptive Flight Control System with Learning System

One way of building a mechanism to allow an adaptive controller to compensate for failed control surfaces and other control system anomalies is to use learning algorithms in the control system design. Although many examples can be found in the literature, this paper concerns the case study of application of the Confidence Tool on a flight testbed at NASA’s Dryden Flight Research Center. The testbed is a specially modified F-15 aircraft (Fig. 3). This aircraft has been modified to include canard control surfaces mounted in the upper inlet area forward of the wing. In test flights, the canards are used to dynamically change the airflow over the wing, thus simulating wing damage.[4] In addition, other normal flight control surfaces (ailerons, stabilators) can also be held fixed to simulate a stuck actuator or loss of a control surface. The aircraft is controlled by a quadruplex, digital, fly-by-wire flight control system that uses a separate computer to implement the neural adaptive controller. With this arrangement, the neural controller can be disabled at any time, leaving a fully functional normal control system to fly the aircraft.

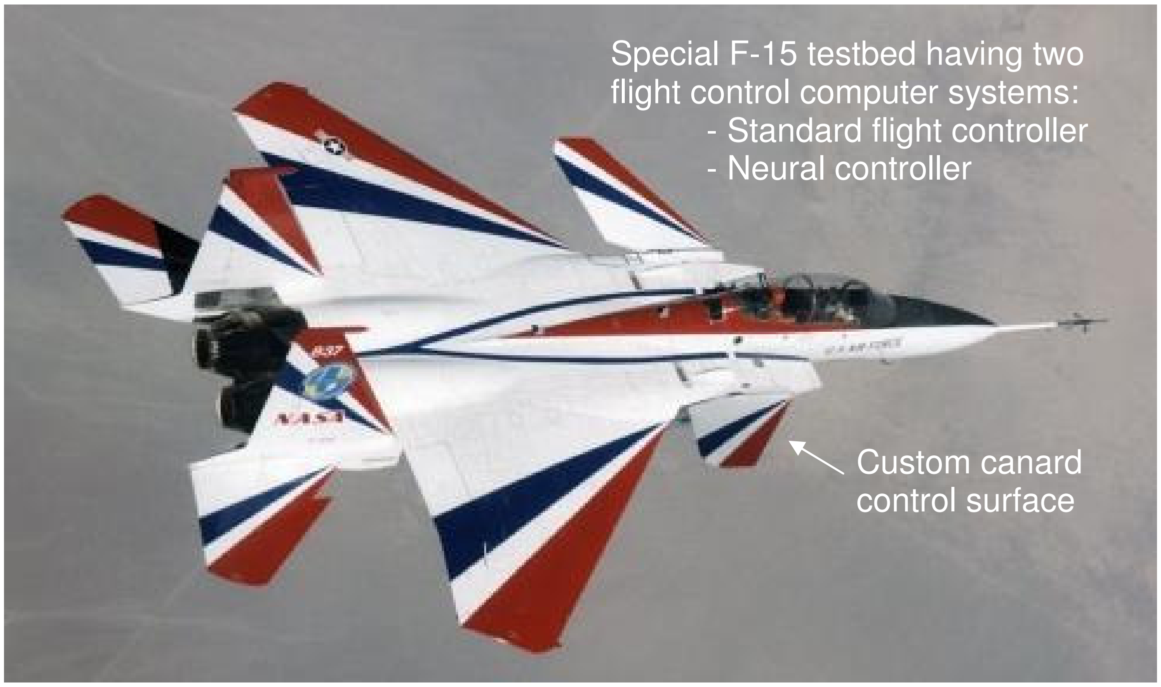
Special F-15 testbed having two flight control computer systems:
- Standard flight controller
- Neural controller
Custom canard control surface

Figure 3: NASA DFRC Intelligent Flight Control System F-15 testbed aircraft.

A neural adaptive controller was designed by NASA Ames Research Center after the manner proposed by Calise and Rysdyk as a means to compensate for aircraft damage in flight.[5] A schematic of the neural adaptive architecture is presented in Fig. 4. Although the scope of this paper does not permit a detailed discussion of the design principles, the details may be found in Refs. 6-7. Looking at just the outermost loop in Fig. 3, it is seen that the measured pitch rate, roll rate, and yaw rate (and other state parameters) are compared to the angular rates commanded by the pilot (rate commands) to form an error signal to drive a PID controller. The PID controller produces intermediate control signals of the type specified by Eq. 2. The inverse plant model takes the PID control signals as input and produces aircraft control surface deflection commands. These commands would be all that is required to control the aircraft were it not for errors in the inverse plant model (potentially as the result of sudden loss of a control surface or aircraft damage).

The function of the neural network is to provide control augmentation commands to help the pilot fly the aircraft in case of damage or modeling errors in the inverse. As indicated in Fig. 3, the neural network takes as inputs the aircraft rate measurements, the augmented control commands being fed to the inverse plant model, and some bias terms. Three neural networks were used for this study; one for pitch augmentation command generation, one for roll augmentation command generation, and one for yaw augmentation command generation. All have the form

$$Augmentation\ \ Command = w^T g \tag{3}$$

where g represent the basis function elements (e.g., pitch-roll-yaw inputs to inverse, scaled body rates, and various products of those terms) and w are the weights of the neural networks. The weights are then updated according to

$$\dot{w} = -k\varepsilon g \tag{4}$$

where the ε term is the error metric fed into the PID controller (Eq. 2). The PID controller gain terms are selected on the basis of a Lyapunov stability analysis.[8-9] The underlying analysis is too complex and detailed to repeat in this paper.

The Neural Network Confidence Tool

It is obvious that the performance of the overall controller strongly depends on the performance of the neural network. If the neural network is not able to learn the required adaptations, it will produce arbitrary outputs that can impede the controller or might even lead to instability. The error ε between the actual and desired behavior drives the learning algorithm. However, a small error does not always imply that the neural network weights are correct. For example, if the pilot is able to learn how to control the aircraft, the

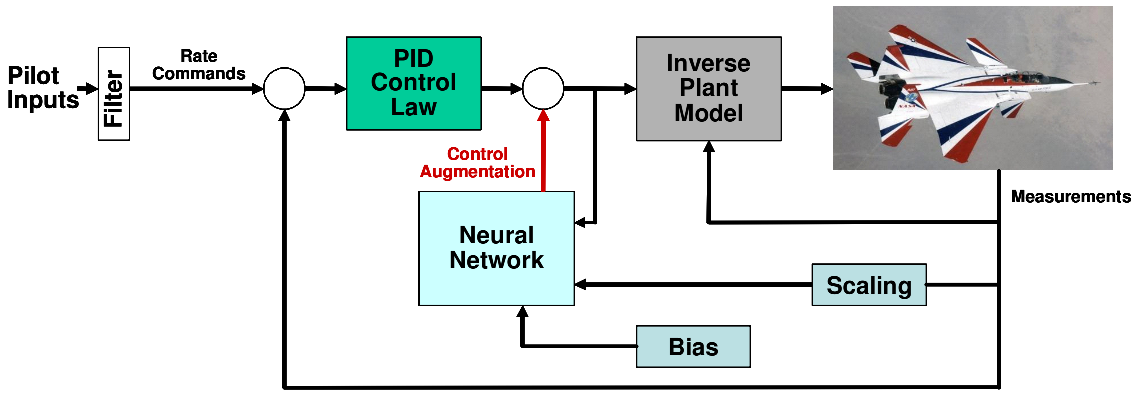

Figure 4. IFCS Damage Adaptive Control System (simplified).

resulting error to drive the learning process effectively becomes shut off. The problem is that if the pilot should lose control, the untrained network may not be able to take over, resulting in a loss of control. Hence, it is important to know when the neural network has not been properly trained, even if the temporary control commands are good.

The Confidence tool is a dynamic monitor, which checks the output values of the neural network and determines if the output of the neural network is reliable by calculating a confidence measure. This metric is based upon a statistical model of the learning system originally developed for pre-trained neural networks, but recently extended for use with on-line learning neural networks. The Confidence tool uses a Bayesian approach to dynamically calculate a confidence measure: an error bar with width σ^2 around each output of the neural network. A small σ indicates good quality, meaning that the output of the neural network is, with a high probability, reasonable given the current weight values. Large σ's indicate poor output estimation quality of the network. Assuming a Gaussian prior p(w) for the neural network weights w, Bayes rule

$$p(w \mid D) = \frac{p(D \mid w)\, p(w)}{P(D)} \tag{5}$$

allows calculation of the effects of learning, i.e., how the weights are influenced by training the network with training data D. This posterior p(w|D) is the probability of the updated weights given the training data D. For the purpose of monitoring network learning, Refs. 10 and 11 show that this calculation can be approximated by

$$\sigma_t^2 = \frac{1}{\beta} + g^T A^{-1} g \tag{6}$$

where g is the gradient of the output with respect to the weight vector, and A is the regularized Hessian matrix computed as

$$A = \frac{\partial(o)}{\partial w_i \partial w_j} + \alpha I \tag{7}$$

I is the identity matrix and α and β are hyper-parameters which are estimated dynamically. Bishop discusses various approaches to estimate the hyper-parameters.[12] This approach is not restricted to the type of neural network used within the IFCS controller, but it can be extended to arbitrary network architectures. The Confidence Tool has been developed to work together with the on-line training algorithm of the IFCS controller. By using a sliding window technique, the performance matrix A^{-1} is approximated without requiring a costly matrix inverse at every step in time. A detailed description of the Confidence Tool can be found in Refs. 10 and 11.

Simulation Results

The Confidence Tool was simulated using a Mathworks Matlab/Simulink model of the controller, neural network, and a nonlinear model of the aircraft dynamics. Figure 5 shows how repeated adaptation in a specific scenario lead to increased confidence in the network's behavior. In the top panel, the roll confidence metric (σ^2) is shown over time (small portion of a simulated test flight). At times *t* = 1, 11, and 17 sec, a doublet commands (fast stick movement from neutral into positive, then negative and back to neutral position) are introduced, as indicated in the lower plot of Fig. 5. At time = 1.5 sec, one control surface of the aircraft (stabilizer) is held fixed to simulate a stuck control surface failure. Because the system dynamics and the model behavior do not match at this point, the neural network produces an augmentation control signal to compensate for this deviation. The network weights are updated according to the given weight update rule. Initially, the network confidence metric is large (meaning low

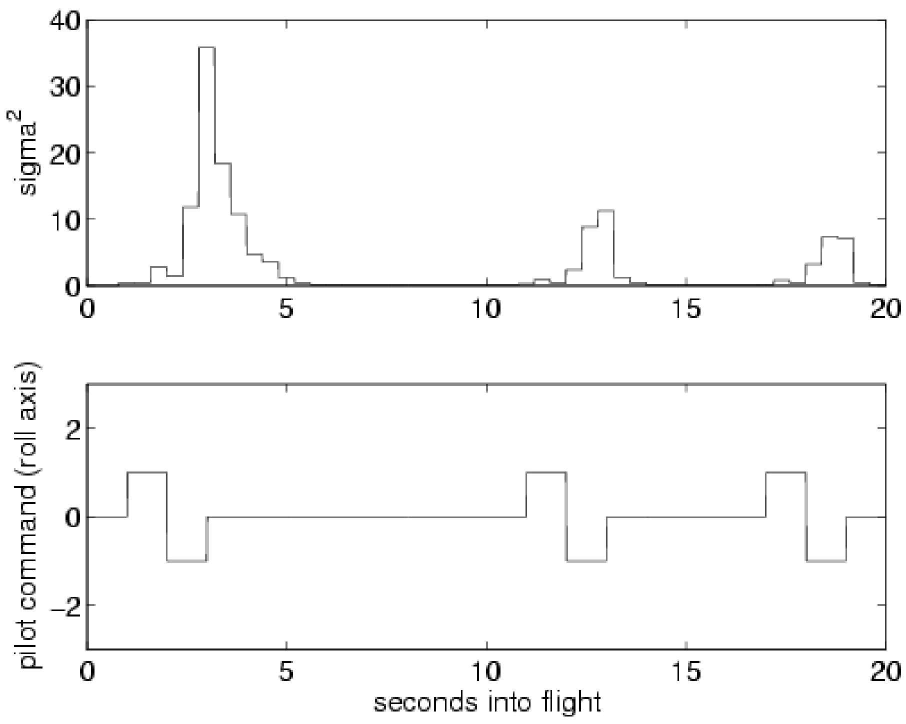

Figure 5: Confidence value s^2 over time (top) and pilot command doublets for roll axis (bottom) during simulation of doublet inputs at times *t* = 1, 11, and 17 sec. A stabilator failure is introduced at *t* = 1.5*s*.

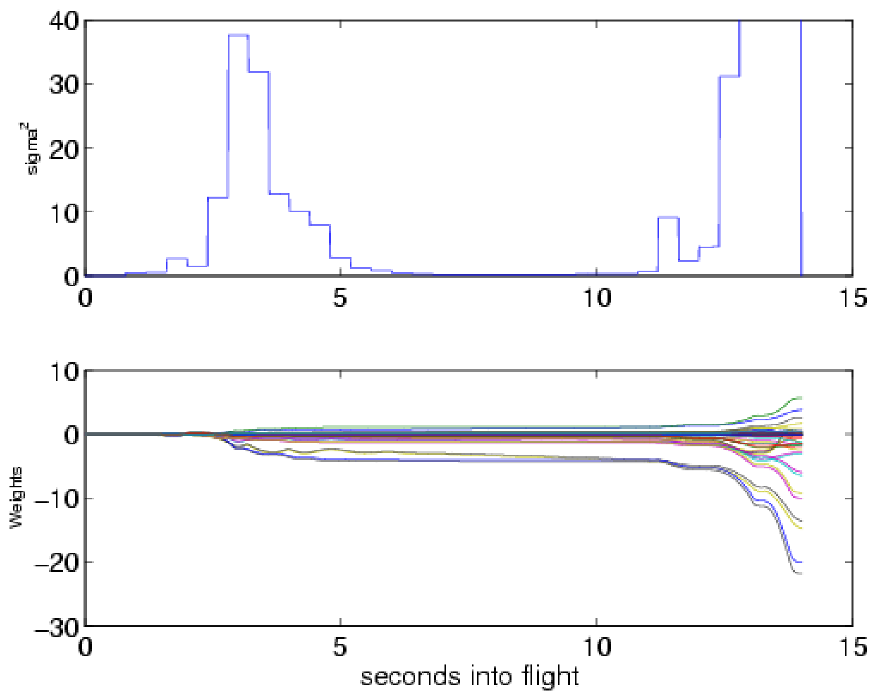

Figure 6: Confidence value σ^2 (top) and neural network weights (bottom) during simulation of doublet inputs at times *t* = 1, 11, and 17 sec with stabilator failure introduced at t = 1.5 sec and corrupted weight update rule.

confidence or indicating a large uncertainty in the network output), but as soon as the damage occurs, the network begins to adapt. Proper learning adaptation is indicated by progressively smaller confidence metrics being generated to subsequent doublet inputs. A small confidence metric indicates a high probability that the network is learning well and that the changes in the neural network outputs with changes in the weight values are small. A second and third doublet input, identical to the first one, are executed at *T*=11*s*, and *T*=17*s*, respectively. During that time, the network's confidence metric is reduced, indicating that the network has successfully adapted to handle the failure situation.

Figure 6 shows the results of a simulation with the same pilot inputs as before, but with the network weight update rule corrupted on purpose to prevent the network from being able to adapt toward a solution. This produced an unstable system, as can be seen by the divergence of the neural network weights in the upper plot after the second pilot command (which also caused termination of the simulator run). The increase of σ^2 during the first doublet is roughly the same as before, although the network takes longer to gain confidence. However, during the second doublet, the weights of the network diverge, leading to large and erratic outputs. The confidence metric (σ^2) reaches a peak value of approximately 140 before the simulation halts. This (albeit extreme) scenario demonstrates the tool's ability to detect—in real time—situations where the neural network is not behaving correctly.

Preliminary Flight Test Data

The Confidence Tool was flight tested on the F-15 IFCS testbed aircraft in the Spring of 2006. During this testing, the neural network controller was evaluated with simulated canard and stabilator failures. A complete exposition of the flight test results must await completion of the flight tests in the Fall of 2006. Nevertheless, the initial data set obtained thus far indicates the Confidence Tool is functional and producing some interesting results. Only a few samples are provided below of the many available to be studied.

Figures 7-10 presents the results obtained from a test point taken at 20,000 feet and 0.75 Mach number while performing a 3G longitudinal tracking maneuver. The objective of this maneuver was to study the research aircraft's (F15) ability to follow a chase aircraft (1000-1500 feet behind) through a 3g turn at constant speed. At time t = 0 sec, the neural net was activated. The left stabilator was "frozen" 7 seconds later to simulate a stuck actuator. The simulated tracking maneuver followed shortly thereafter. Each figure has four sub-plots: a plot of the stabilator positions to show the introduction of failure, the weight values of the pitch neural network, the pitch neural network confidence metric, and finally the pitch portion of the PID error signals.

Figures 7 and 8 show the effect of a -2 deg failure of the left stabilator. The only difference in the test conditions for the data shown in these two figures was that in Fig. 8 the neural network was given training signal inputs. These were generated as pitch doublets introduced by the pilot during approximately the first 50 seconds of the record. In Fig. 7, the confidence metric becomes larger (worse confidence) during the periods at which one weight is especially seen to demonstrate convergence problems. The bottom-most plot of Fig. 7 shows significant controller errors occurred during those periods. In contrast, the bottom plot of Fig. 8 indicates that the training inputs given to the network appear to have improved the neural network learning, as desired. In Fig. 8, from t = 20 to t = 40 where after the doublet input ceases, the control error is seen to be driven nearly to zero and the neural network values do not appear to be changing significantly. Although the true value of the network weights is not known, the fact that the control error signals are low indicates that convergence to the correct weights is highly probable. Also during this period, it is seen that the Confidence metric drops nearly to zero, indicating very good confidence in the network.

Figures 9 and 10 show the effect of a larger, -4 deg failure of the left stabilator. Again, the only difference in the test conditions for the data shown in these two figures was that in Fig. 10 the neural network was given training input signals (pitch doublets) during approximately the first 50 seconds. Comparing the lower-most subplots of Figs. 9 and 10, it is seen that the training inputs again likely helped to lower the

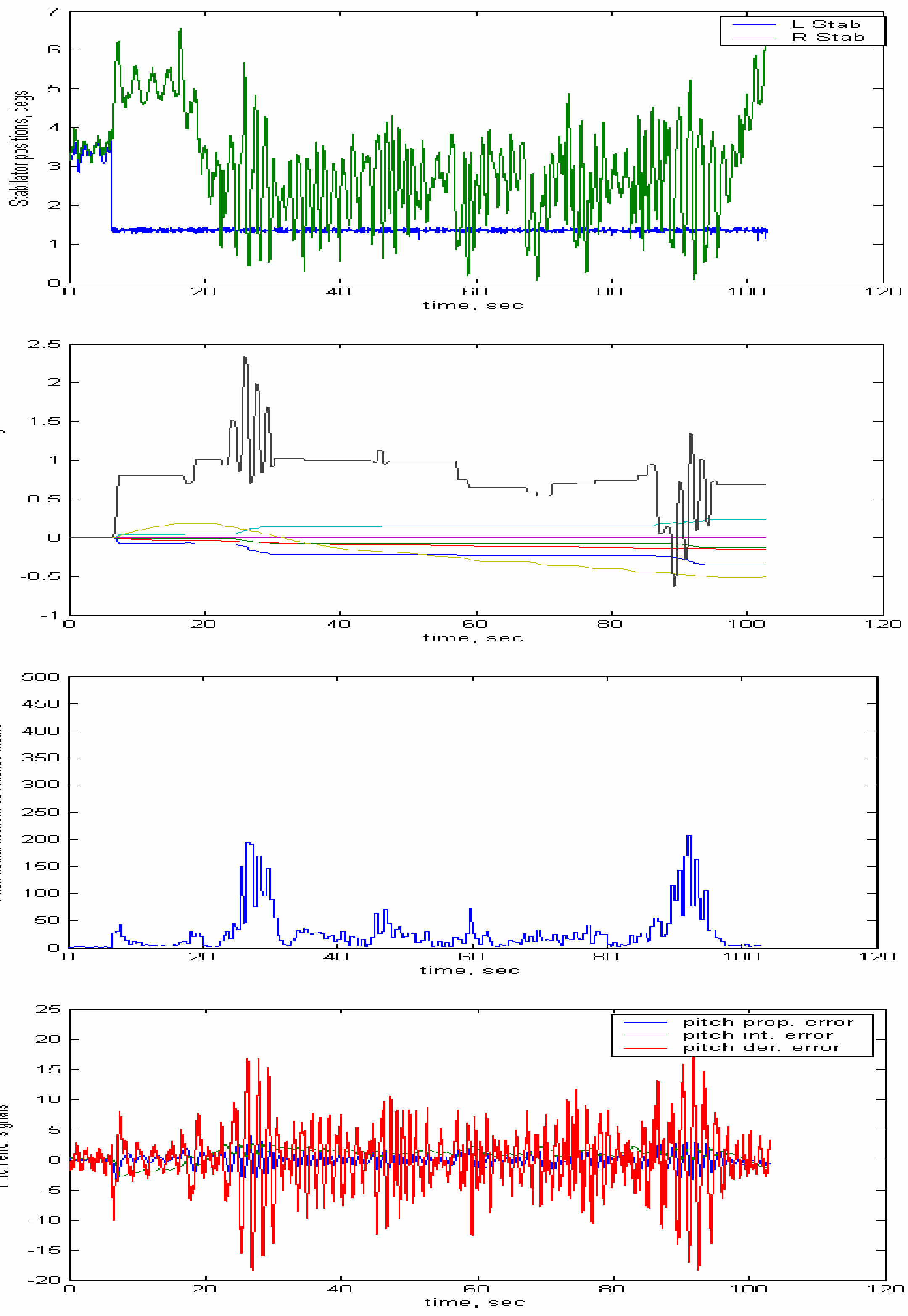

Figure 7: Flight test data showing effect of -2 deg locked stabilator at *t* = 7 sec on pitch neural network weights, pitch confidence metric, and pitch controller error signals during a 3-g tracking maneuver at 20,000 ft and 0.75 Mach number. Run 193 csv 1.

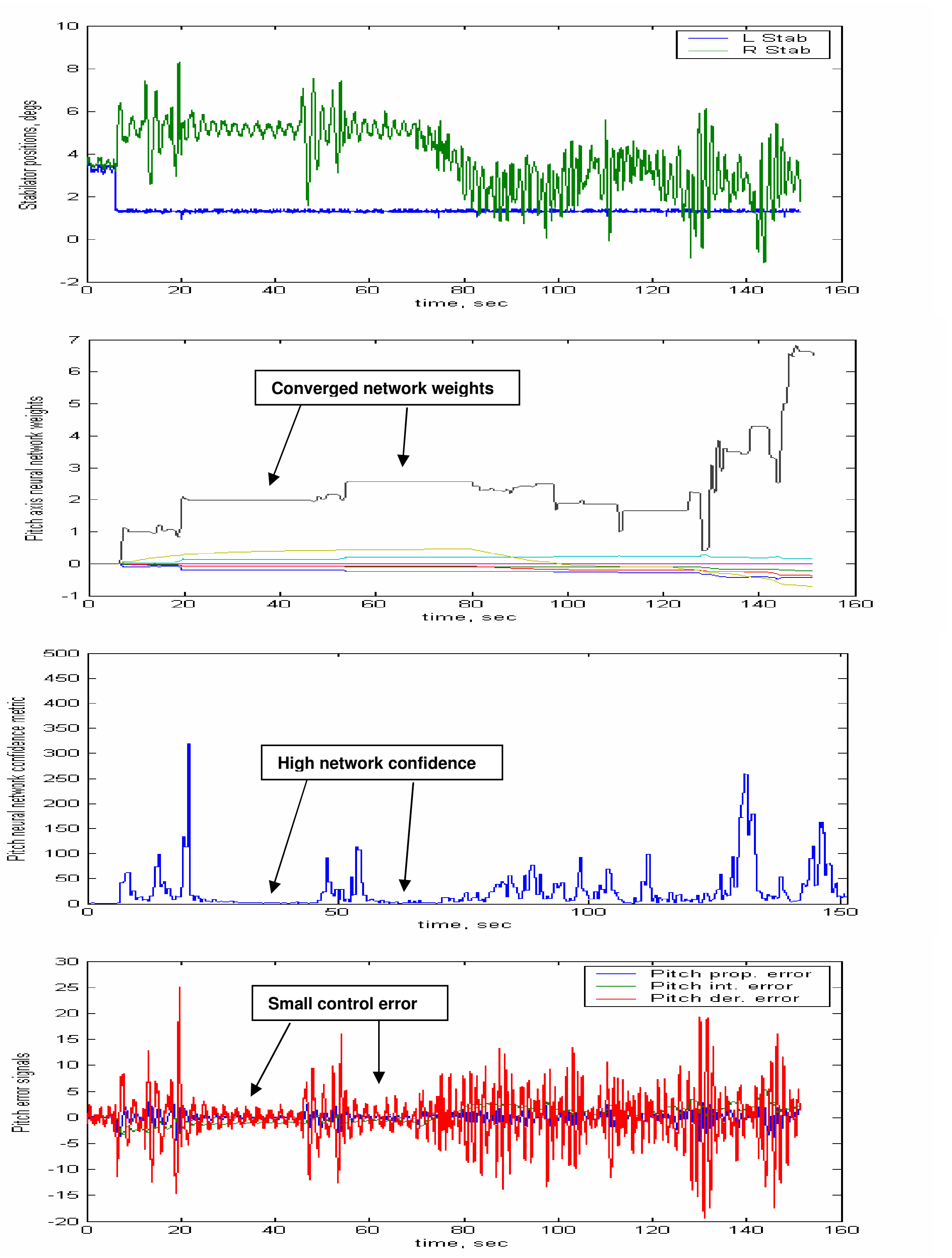

Figure 8: Same test condition as Fig. 7 but with pitch doublets introduced at t = 15, 20, 45, and 55 sec, then followed by 3g tracking maneuver. Run 193 csv 2.

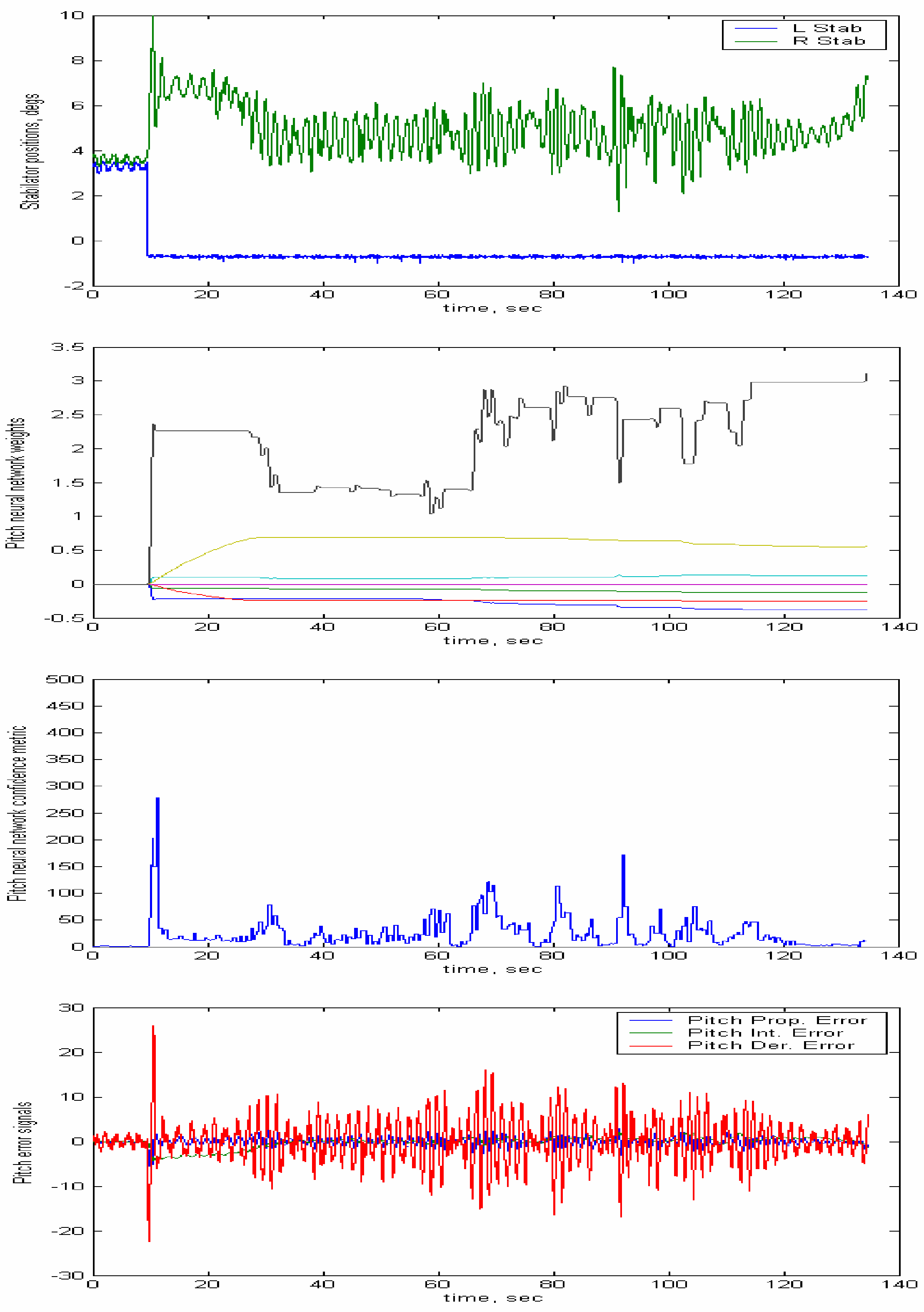

Figure 9: Flight test data showing effect of -4 deg locked stabilator at *t* = 7 sec on pitch neural network weights, pitch confidence metric, and pitch controller error signals during a 3-g tracking maneuver at 20,000 ft and 0.75 Mach number. Run 193 csv 1.

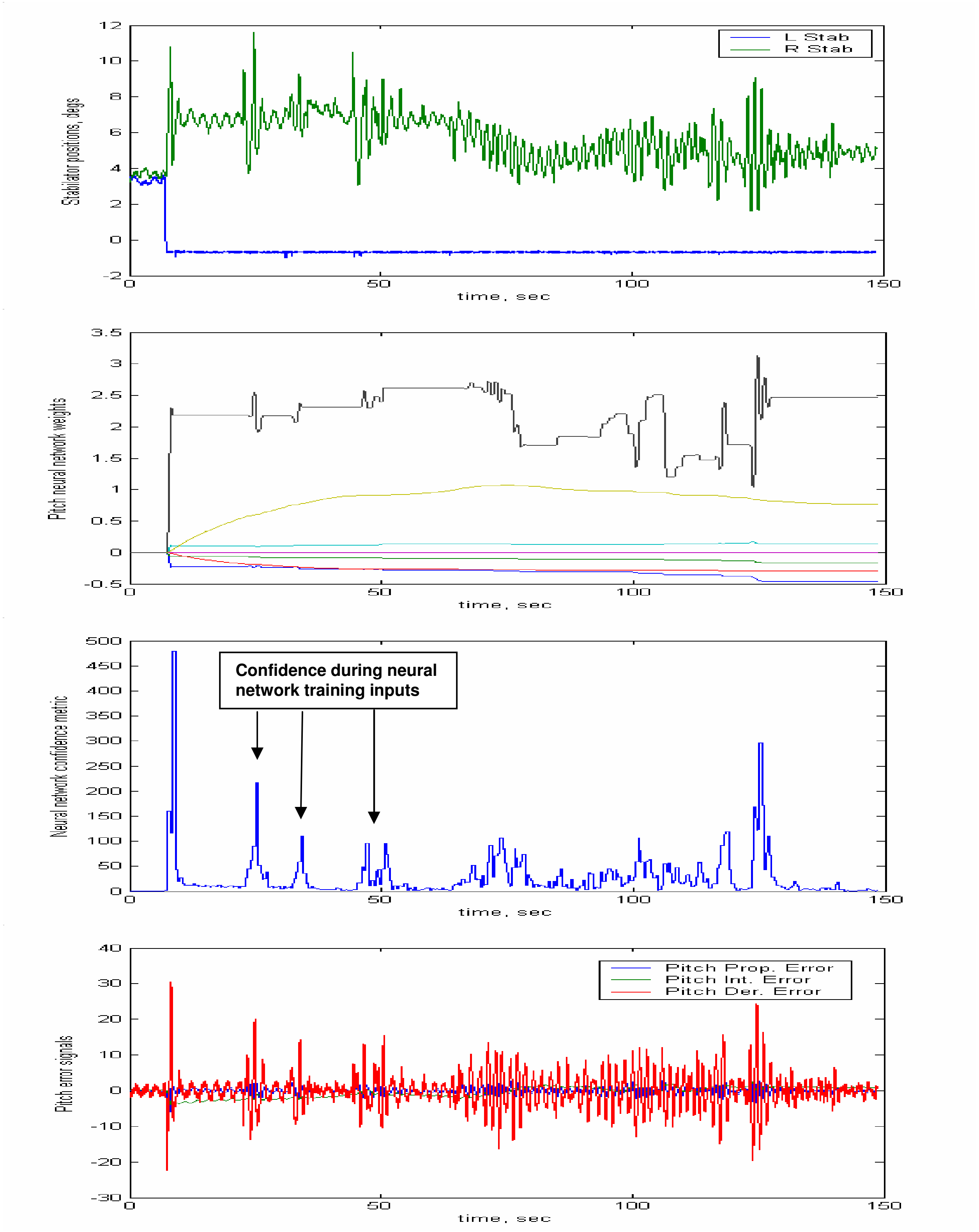

Figure 10: Same test condition as Fig. 9 but with pitch doublets introduced at t = 20, 30, and 45 sec, then followed by 3g tracking maneuver. Run 193 csv 4.

error in the neural network weights. In examining the control errors for the first 60 seconds, it is seen that after the pilot stops the doublet training inputs, the resulting control errors are much less (Fig. 10) in comparison to the same time period without training (Fig. 9). This indicates an improvement in the handling qualities might be noticed, and in fact did appear to agree with pilot assessments. Figure 10 also shows progressively lower confidence metric values (better learning) with successive doublet inputs. This is reminiscent of the simulation findings shown earlier in Fig. 5. It indicates that the neural network weights have progressively moved closer to their true values.

For each test point, the pilots were asked to evaluate the handling qualities according to the Cooper-Harper rating (CHR) scale, whereby a rating of 1 is the best, and 10 is the worst. Table 1 presents the CHR ratings for the four cases shown in Figs. 7-10 in comparison to the baseline cases.[13] There were two baselines: 1) baseline handling quality with no stabilator failure, and 2) handling quality with stabilator failure, but with the neural networks turned off. The pilots rated the aircraft for gross target acquisition, fine tracking, and pitch-induced oscillation (PIO) tendency. In all cases shown in Table 1, the neural networks did not restore the same handling quality ratings to the baseline (no failure) level. Interestinglly, the training inputs used to improve neural network learning appeared to have reduced the handling qualities for the case of -2 deg failure, yet improved the handling qualities for the case of -4 deg failure. This phenomena remains under investigation. At the test points shown in Table 1, the neural networks induced considerable PIO effects not seen in the baseline aircraft.

Although the neural networks did not fully restore the baseline handling qualities during failure, this does not mean that the neural networks were not successfully employed. The failures tested in this flight test program were necessarily benign to maximize pilot safety. It is very possible that given more severe failure situations, the neural networks might be able to give an otherwise uncontrollable aircraft enough control authority to land, albeit with somewhat degraded handling qualities.

Table 1: Comparison of Handling Qualities Using the Cooper-Harper Rating Scale

Stab Failure Magnitude	Gross Acquisition CHR	Fine Tracking CHR	PIO rating
Baseline (flight 193)	2	2	1
-2 deg from trim; NN off	4	3	1
-2 deg from trim; NN on	3	3	4
-2 deg from trim; NN on with training	5	4	4
-4 deg from trim; NN off	5	4	4
-4 deg from trim; NN on	4	5	4
-4 deg from trim; NN on with training	3	3	4

Conclusions

Given the both the incomplete understanding of the test data that has been so recently acquired and that more data will likely soon be acquired, it is too early to make conclusive statements regarding the ability of the neural-adaptive controller to restore aircraft performance in the event of failure. The data does show, however, that the Confidence Tool appears to be able to calculate a useful metric of neural network learning performance. As such, the output of the Confidence Tool might be suitable to queue controller actions or to produce pilot advisory warnings which could play a significant role towards the eventual certification of neural-adaptive controllers.

References:

[1] Kaneshige, J., Bull, J., and Totah, J., "Generic Neural Flight Control and Autopilot System," AIAA-2000-4281.

[2] Rysdyk, R. T., and Calise, A.J., " Fault tolerant flight control via Adaptive Neural Network Augmentation," AIAA 98-4483, August 1998.

[3] Gupta, P. and Schumann, J., "A Tool for Verification and Validation of Neural Network Based Adaptive Controllers for High Assurance Systems", In Proceedings High Assurance Software Engineering (HASE). IEEE, 2004

[4] Williams-Hayes, Peggy S., “Flight Test Implementation of a Second Generation Intelligent Flight Control System,” NASA/TM-2005-213669, November 2005.

[5] Calise, A. J., and Rysdyk, R. T., “Adaptive Model Inversion Flight Control for Tiltrotor Aircraft,” Proceedings of AIAA Guidance, Navigation, and Control Conference, Aug. 1997, Paper No. 97-3758.

[6] Kaneshige, J., Bull, J., and Totah, J. J., “Generic Neural Flight Control and Autopilot System,” AIAA-2000-4281, August 2000.

[7] Kaneshige, J. and Gundy–Burlet, K., “Integrated Neural Flight and Propulsion Control System,” AIAA 2001-4386, August 2001.

[8] Leitner, J., Calise, A., and Prasad, J. V. R., “Analysis of Adaptive Neural Networks for Helicopter Flight Control,” Proceedings of the AIAA Guidance, Navigation, and Control Conference, Baltimore, MD, 1995.

[9] Kim, B. S., and Calise, A., “Nonlinear Flight Control Using Neural Networks,” Proceedings of the AIAA Guidance, Navigation, and Control Conference, Scottsdale, AZ, 1994.

[10] Schumann, J., and Gupta, P., “Monitoring the Performance of a Neuro-Adaptive Controller,” Proceedings of MAXENT 2004, AIP Conference Proceedings 735, pp 289–296, American Institute of Physics, 2004.

[11] Gupta, P., Guenther, K., Hodgkinson, J., Jacklin, S., Richard, M., Schumann, J., and Soares, F., “Performance Monitoring and Assessment of Neuro-Adaptive Controllers for Aerospace Applications Using a Bayesian Approach,” Proceedings of the AIAA Guidance, Navigation, and Control Conference, San Francisco, CA, 2005.

[12] Bishop, C. M., *Neural Networks for Pattern Recognition*, Claderon Press, Oxford, 1995.

[13] Williams-Hayes, Peggy S., and Bosworth, J. T., “Flight Test Results for an F-15 Aircraft with Simulated Asymmetric Surface Failures Using Neural Network Adaptation,” NASA TM (number TBD), submitted June 21, 2006.